We claim:

1. A method for using a universal client program to operate an application program executing on a server system, comprising:

launching a universal client program having a synchronized state with an application program associated with a server;

exchanging at least one protocol message between the universal client program and the application program; wherein at least one of the protocol messages indicates an update to the user interface of the client; and

updating the user interface of the client from the update to the user interface of the client.

2. The method as recited in claim 1, further comprising:

maintaining a first object-oriented hierarchy representing user interface state by the universal client program; and

synchronizing dynamically the first object-oriented hierarchy and a second object-oriented hierarchy representing user interface state by the application program by means of an event-driven synchronization protocol.

3. A computer-readable medium having computer-executable instructions to cause a client computer to perform a method comprising:

launching a universal client program having a synchronized state with an application program associated with a server;

exchanging at least one protocol message between the universal client program and the application program; wherein at least one of the protocol messages indicates an update to the user interface of the client; and

updating the user interface of the client from the update to the user interface of the client.

4. The computer-readable medium as recited in claim 3, further comprising:

maintaining a first object-oriented hierarchy representing user interface state by the universal client program; and

synchronizing dynamically the first object-oriented hierarchy and a second object-oriented hierarchy representing user interface state by the application program by means of an event-driven synchronization protocol.

5. A computer data signal embodied in a carrier wave and representing a sequence of instructions which, when executed by a processor, cause the processor to perform the method of:

launching a universal client program having a synchronized state with an application program associated with a server;

exchanging at least one protocol message between the universal client program and the application program; wherein at least one of the protocol messages indicates an update to the user interface of the client; and

updating the user interface of the client from the update to the user interface of the client.

6. The computer data signal as recited in claim 5, further comprising:

maintaining a first object-oriented hierarchy representing user interface state by the universal client program; and

synchronizing dynamically the first object-oriented hierarchy and a second object-oriented hierarchy representing user interface state by the application program by means of an event-driven synchronization protocol.

7. A method for using a universal client program to operate an application program executing on a server system, comprising:

establishing communication between the universal client program associated with a client and an application program associated with a server; and

operating the application program by presenting the user interface of the application program using the universal client program.

8. The method as recited in claim 7, wherein establishing communication between the universal client program and the application program further comprises:

establishing communication between the client and the server

sending a message from the universal client program to the server program, the message requesting access to the application program;

executing the requested application program upon the server; and

placing the universal client program in communication with the requested application program.

9. The method as recited in claim 7, wherein operating the application program by presenting the user interface of the application program using the universal client program further comprises:

rendering the user interface of the application program on the client by the universal client program;

maintaining a first object-oriented hierarchy representing user interface state by the universal client program; and

synchronizing dynamically the first object-oriented hierarchy and a second object-oriented hierarchy representing user interface state by the application program by means of an event-driven synchronization protocol.

10. The method as recited in claim 9, wherein synchronizing dynamically the object-oriented hierarchies recited further comprises:

executing corresponding universal object logic by the universal client program in response to user input events, independently of communication with the application program;

communicating user input events from the universal client program to the application program; and

communicating user interface updates from the application program to the universal client software, excluding updates known by the application program to have already been performed independently by the universal client program based on a shared knowledge of universal object logic.

11. The method as recited in claim 9, wherein the event-driven synchronization protocol contains no platform specific information, wherein the universal client program, as adapted to some particular client hardware platform, operates any application program that is adapted to execute on a particular server hardware platform.

12. The method as recited in claim 9, wherein the event-driven synchronization protocol contains no executable code segments, such that the universal client program is readily adaptable to any particular client hardware platform without additionally adapting an execution framework selected from a list consisting of a virtual machine, and a scripting language interpreter.

13. The method as recited in claim 12, wherein the event-driven synchronization protocol is encrypted.

14. The method as recited in claim 7, wherein network bandwidth usage is minimized with respect to a number of network messages exchanged between the universal client program and the application program by using shared knowledge of universal object logic.

15. The method as recited in claim 7, wherein the universal client program, the server program, and the application program exist on a same physical computer.

16. The method as recited in claim 7, wherein the universal client program and the application program exist on computer hardware systems connected by means of cabling including electronic and fiber optic.

17. The method as recited in claim 7, wherein the universal client program and the application program exist on computer hardware systems connected by wireless means.

18. The method as recited in claim 7, wherein a general network protocol comprises the Transmission Control Protocol/Internet Protocol.

19. A system for transacting in electronic commerce comprising:

a processor; and

means operative on the processor for establishing communication between a universal client program and an application program; and operating the application program by presenting the user interface of the application program using the universal client program.

20. A computer-readable medium having computer-executable instructions to cause a client computer to perform a method comprising:

establishing communication between a universal client program and an application program; and

operating the application program by presenting the user interface of the application program using the universal client program.

21. The computer-readable medium as recited in claim 20, wherein establishing communication between the universal client program and the application program further comprises:

establishing communication between the client and the server

sending a message from the universal client program to the server program, the message requesting access to the application program;

executing the requested application program upon the server; and

placing the universal client program in communication with the requested application program.

22. The computer-readable medium as recited in claim 20, wherein operating the application program by presenting the user interface of the application program using the universal client program further comprises:

rendering the user interface of the application program on the client by the universal client program;

maintaining a first object-oriented hierarchy representing user interface state by the universal client program; and

synchronizing dynamically the first object-oriented hierarchy and a second object-oriented hierarchy representing user interface state by the application program by means of an event-driven synchronization protocol.

23. The computer-readable medium as recited in claim 22, wherein synchronizing dynamically the object-oriented hierarchies recited further comprises:

executing corresponding universal object logic by the universal client program in response to user input events, independently of communication with the application program;

communicating user input events from the universal client program to the application program; and

communicating user interface updates from the application program to the universal client software, excluding updates known by the application program to have already been performed independently by the universal client program based on a shared knowledge of universal object logic.

24. The computer-readable medium as recited in claim 22, wherein the event-driven synchronization protocol contains no platform specific information, wherein the universal client program, as adapted to some particular client hardware platform, operates any application program that is adapted to execute on a particular server hardware platform.

25. The computer-readable medium as recited in claim 22, wherein the event-driven synchronization protocol contains no executable code segments, such that the universal client program is readily adaptable to any particular client hardware platform without additionally adapting an execution framework selected from a list consisting of a virtual machine, and a scripting language interpreter.

26. The computer-readable medium as recited in claim 25, wherein the event-driven synchronization protocol is encrypted.

27. A computer data signal embodied in a carrier wave and representing a sequence of instructions which, when executed by a processor, cause the processor to perform the method of:

establishing communication between a universal client program and an application program; and

operating the application program by presenting the user interface of the application program using the universal client program.

28. A client method for using a universal client program to operate an application program executing on a server system, comprising:

establishing communication between the universal client program and an application program;

operating the application program by presenting the user interface of the application program using the universal client program,

wherein the event-driven synchronization protocol contains no platform specific information;

wherein the universal client program, as adapted to a client hardware platform, operates any application program that is adapted to execute on a server hardware platform,

wherein operating the application program by presenting the user interface of the application program using the universal client program further comprises:

rendering the user interface of the application program on the client system by the universal client program;

maintaining a first object-oriented hierarchy representing user interface state by the universal client program;

maintaining a second object-oriented hierarchy representing user interface state by the application program; and

synchronizing dynamically the first object-oriented hierarchy and the second object-oriented hierarchy by means of an event-driven synchronization protocol.

29. The method as recited in claim 28, wherein synchronizing dynamically the object-oriented hierarchies further comprises:

executing corresponding universal object logic by the universal client program in response to user input events, independently of communication with the application program;

communicating user input events from the universal client program to the application program; and

communicating user interface updates from the application program to the universal client software, excluding updates known by the application program to have already been performed independently by the universal client program based on a shared knowledge of universal object logic.

30. A system for transacting in electronic commerce comprising:

a processor;

a universal client program operative on the processor;

an application program operative on the processor;

software means operative on the processor for establishing communication between the universal client program and the application program; and

software means for operating the application program by presenting the user interface of the application program using the universal client program.

31. A computer-readable medium having computer-executable instructions to cause a client computer to perform a method comprising:

establishing communication between the universal client program and an application program;

operating the application program by presenting the user interface of the application program using the universal client program;

wherein operating the application program by presenting the user interface of the application program using the universal client program further comprises:

rendering the user interface of the application program on the client system by the universal client program;

maintaining a first object-oriented hierarchy representing user interface state by the universal client program;

maintaining a second object-oriented hierarchy representing user interface state by the application program; and

synchronizing dynamically the first object-oriented hierarchy and the second object-oriented hierarchy by means of an event-driven synchronization protocol;

wherein the event-driven synchronization protocol contains no platform specific information;

wherein the universal client program, as adapted to a client hardware platform, operates any application program that is adapted to execute on a server hardware platform.

32. A computer data signal embodied in a carrier wave and representing a sequence of instructions which, when executed by a processor, cause the processor to perform the method of:

establishing communication between the universal client program and an application program;

operating the application program by presenting the user interface of the application program using the universal client program;

wherein operating the application program by presenting the user interface of the application program using the universal client program further comprises:

rendering the user interface of the application program on the client system by the universal client program;

maintaining a first object-oriented hierarchy representing user interface state by the universal client program;

maintaining a second object-oriented hierarchy representing user interface state by the application program; and

synchronizing dynamically the first object-oriented hierarchy and the second object-oriented hierarchy by means of an event-driven synchronization protocol;

wherein the event-driven synchronization protocol contains no platform specific information;

wherein the universal client program, as adapted to a client hardware platform, operates any application program that is adapted to execute on a server hardware platform.

33. A method for using a single instance of a universal client program to operate a plurality of application programs executing on a plurality of server systems, comprising:

with respect to the operation of a single application, the method comprising

establishing communication between the universal client program and an application program; and

operating the application program by presenting the user interface of the application program using the universal client program;

with respect to managing a plurality of operations, the method comprising further comprising:

sending messages to server programs requesting access to additional application programs, which are to be executed upon other server systems;

executing the additional requested application programs on these other server systems and placing additional requested application programs in communication with the universal client program; and

switching between applications such that at any given moment one particular application program is distinguished as the active application, such that during this time the active application handles all user input events.

34. The method as recited in claim 33, wherein establishing communication between a universal client program and the application program further comprises:

establishing communication between a client system and a server system;

sending a message from the universal client program to a server program, the message requesting access to an application program, which is to be executed upon the server system;

executing the requested application program upon the server system; and

placing the universal client program in communication with the requested application program.

35. The method as recited in claim 32, wherein operating the application program by presenting the user interface of the application program using the universal client program further comprises:

rendering the user interface of the application program on the client system by the universal client program;

maintaining a first object-oriented hierarchy representing user interface state by the universal client program;

maintaining a second object-oriented hierarchy representing user interface state by the application program; and

synchronizing dynamically the first object-oriented hierarchy and the second object-oriented hierarchy by means of an event-driven synchronization protocol.

36. The method as recited in claim 35, wherein synchronizing dynamically the object-oriented hierarchies recited further comprises:

executing appropriate universal object logic by the universal client program in response to user input events, independently of communication with the application program;

communicating user input events from the universal client program to the application program; and

communicating user interface updates from the application program to the universal client software, excluding any updates known by the application program to

have already been performed independently by the universal client program based on a shared knowledge of universal object logic.

37. A method for using a universal client program to operate an application program executing on a server system, comprising:

sending locally-generated protocol events to the server via protocol messages;

responding to server-generated protocol events received as protocol messages; and

maintaining a shared state of the application executing on the server.

38. A client system for transacting in electronic commerce comprising:

a processor; and

means operative on the processor for sending locally-generated protocol events to a server via protocol messages, responding to server-generated protocol events received as protocol messages, and maintaining a shared state of an application executing on the server.

39. A computer-readable medium having computer-executable instructions to cause a client computer to perform a method comprising:

sending locally-generated protocol events to a server via protocol messages;

responding to server-generated protocol events received as protocol messages; and

maintaining a shared state of an application executing on the server.

40. A computer data signal embodied in a carrier wave and representing a sequence of instructions which, when executed by a processor, cause the processor to perform the method of:

sending locally-generated protocol events to a server via protocol messages;

responding to server-generated protocol events received as protocol messages; and

maintaining a shared state of an application executing on the server.

41. A computerized apparatus, comprising:

a plurality of objects comprising an object-oriented class hierarchy, the hierarchy defining a user interface of an application; and

an application operably coupled to at least one of the plurality of objects, wherein the at least one object defines the user interface of the application, and the application is operated by a universal client program,

wherein the at least one object has access to the context in which the at least object is operably coupled,

wherein the at least one object has access to information indicating when to transmit and receive an event-driven synchronization protocol message,

wherein invocation of each method of the at least one object is performed without regard to the details of a managing synchronization implementation, which are encapsulated within each method.

42. A computerized apparatus as recited as in claim 41, wherein the context further comprises:

a synchronized object hierarchy of either the universal client program or the application program.

43. A computerized apparatus for remote client computing, comprising:

a universal client program executing on the client; and

a dynamic object-oriented hierarchy representing a user interface state associated with the universal client program,

wherein the client is operably coupled through a communication network to a server executing an application program, enabling the universal client program to communicate to the application program,

wherein the communication network implements a general network protocol for communicating digital information over the communication network,

wherein the object-oriented hierarchies are synchronized dynamically using an event-driven synchronization protocol.

44. A computerized server apparatus for remote computing, comprising:

at least one application program resident in a persistent memory of the server system;

a dynamic object-oriented hierarchy representing a user interface state associated with the application program;

wherein the server is operably coupled through a communication network to a client executing an universal client program, enabling the application program to communicate to the universal client program,

wherein the communication network implements a general network protocol for communicating digital information over the communication network,

wherein the object-oriented hierarchies are synchronized dynamically using an event-driven synchronization protocol.

45. A computer-readable medium comprising:

(a) a universal client program; and

(b) a dynamic object hierarchy representing user interface state associated with the universal client program.

46. A computer-readable medium comprising:

(a) a server program;

(b) an application program operably coupled to the server program; and

(c) a dynamic object-oriented hierarchy representing user interface state associated with the application program.

47. A system for transacting in electronic commerce comprising:

a processor; and

software means operative on the processor for representing to a dynamic object hierarchy, a user interface state associated with a universal client program.

48. A method for using a universal client program to operate an application program executing on a server system, comprising:

launching a universal client;

managing a link request associated with the universal client; and

operating an application program on a server, through the universal client.

49. The method as recited in claim 48, wherein launching a universal client further comprises:

instantiating a root object;

instantiating a local link request application object;

creating a hierarchical relationship between the root object and the local link request application object; and

enabling acceptance of user input associated with the application object.

50. The method as recited in claim 49, wherein enabling acceptance of user input further comprises:

passing control to the local link request application object.

51. The method as recited in claim 49,

wherein the root object is instantiated by a startup logic component;

wherein the local link request application object is instantiated by the startup logic component; and

wherein the hierarchical relationship is created by the startup logic component;

52. The method as recited in claim 48, wherein managing a link request further comprises:

receiving a server address and an identification of the application program;

instantiating an application object as a member of the universal client program;

instantiating a node as a member of the application object;
instantiating a socket as a member of the node;
instantiating a disabled cipher object;
instantiating a disabled decipher object ;
parsing an IP address of the server from the server address; and
enabling the client to monitor incoming connection requests using the IP address.

53. The method as recited in claim 52, wherein the application object is instantiated by the link request application.

54. The method as recited in claim 52, wherein the cipher object and the decipher object are instantiated each as a member of the node.

55. The method as recited in claim 52, wherein enabling the client to monitor incoming connection requests further comprises establishing a communication path between the client socket and a server socket using the IP address.

56. The method as recited in claim 55, wherein the communication path further comprise a TCP/IP connection.

57. The method as recited in claim 55, wherein the communication path further comprise a UDP/IP connection.

58. The method as recited in claim 55, wherein the method further comprises:
affirming the establishment of the communication path.

59. The method as recited in claim 52, wherein the method further comprises:
receiving a connect message from the server;
invoking a connect protocol event within the universal client program; and
sending a link request protocol message to the server.

60. The method as recited in claim 48, wherein operating an application program through the universal client further comprises:

creating at least one graphical user interface object;

modifying at least one attribute of the at least one graphical user interface object; and

managing at least one click event of the at least one graphical user interface object.

61. The method as recited in claim 60, wherein managing at least one click event of the at least one graphical user interface object further comprises:

receiving a protocol event;

wherein a local input handler registers the protocol event;

determining which graphical user interface object is the target of the protocol event;

triggering a local protocol event for the target graphical user interface object; and

managing the local protocol event.

62. The method as recited in claim 61, wherein managing the local protocol event further comprises:

producing appropriate visual feedback effects, as universally defined for the type of the target graphical user interface object; and

encoding a click protocol message, including coordinates of the click event and an identification of the target graphical user interface object.

63. The method as recited in claim 60, wherein creating at least one graphical user interface object further comprises:

receiving a message to create a graphical user interface object;

decoding at least one parameter and at least one attribute associated with the message;

instantiating a graphical user interface object within the hierarchical relationship associated with the application object;

placing the graphical user interface object in a hierarchical relationship to the application object; and

initializing the at least one attribute of the graphical user interface object with the at least one parameter and at least one attribute associated with the message.

64. The method as recited in claim 63,

wherein the message further comprises a create visual object protocol message;

65. The method as recited in claim 60, wherein modifying attributes of the at least one graphical user interface object further comprises:

receiving a new attribute value;

comparing the new attribute value to an old attribute value;

setting the attribute value to the new attribute value;

encoding a change attribute value protocol message with the identification of the graphical user interface object, the identification of the attribute and the new attribute value; and

transmitting the protocol message,

wherein the setting, encoding, and the transmitting are performed when the comparing indicates inequality.

66. The method as recited in claim 65, wherein the method further comprises:

receiving the change attribute value protocol message; and

modifying the value of the specified attribute in the specified object to the specified new value;

wherein the modifying is performed by the universal object logic in the universal client program

67. The method as recited in claim 66, wherein the method further comprises:

receiving the change attribute value protocol message;

encoding a second change attribute value protocol message; and

transmitting second change attribute value protocol message,

wherein the receiving, the encoding and the transmitting is performed by the universal client program.

68. A method for enabling operation by a client of an application program executing on a server system, comprising:

launching the application program;

managing link requests; and

managing the application program.

69. The method as recited in claim 68, wherein launching the application program further comprises:

instantiating an application object;

instantiating an application manager as a member of the application object;

instantiating a launch manager as a member of the application object;

instantiating a node as a member of the application object;

instantiated a disabled cipher object and a disabled decipher object as members of the node;

instantiating a socket as a member of the node;

instantiating an administrative application object; and

associating a thread with the socket.

70. The method as recited in claim 68, wherein managing link requests further comprises:

affirming the communication path with the client, from URL information;

parsing the identification of the application program from the URL information; and

determining availability of the application program on the server.

71. The method as recited in claim 70, wherein the determining indicates that the application program is not available on the server, the method further comprises:

informing the client of the server unavailability.

72. The method as recited in claim 70, wherein the determining indicates that the application program is available on the server, the method further comprises:

informing a load balancer of the request; and

waiting for a response from the load balancer.

73. The method as recited in claim 72, wherein the informing the load balancer of the request further comprises:

sending a launch-approved message to the launch manager.

74. The method as recited in claim 70, wherein affirming the communication path further comprises:

sending an ST connect message to the client;

receiving a link request from the client; and

generating a link request protocol event.

75. The method as recited in claim 70, wherein the link request includes URL information.

76. The method as recited in claim 70, wherein determining availability of the application program on the server further comprises:

determining whether or not server resources are fully utilized.

77. The method as recited in claim 76, wherein the determining resolves an indication of less than fully utilized server resources, the method further comprises:

sending a launch-approved message to the client.

78. The method as recited in claim 76, wherein the determining resolves an indication that server resources are fully utilized, the method further comprises:

determining whether or not auxiliary server resources are available.

79. The method as recited in claim 78, wherein auxiliary server resources are determined to be not available, the method further comprises:

sending a denial of service message to the client.

80. The method as recited in claim 78, wherein auxiliary server resources are determined to be available, the method further comprises:

selecting an auxiliary server; and

sending a message indicating the selected auxiliary server to the client.

81. The method as recited in claim 68, wherein managing the application program further comprises:

creating a graphical user interface object;

modifying at least one attributes of the graphical user interface object; and

managing click events of the graphical user interface object.

82. The method as recited in claim 81, wherein creating a graphical user interface object further comprises:

encoding a protocol message indicating creation of the graphical user interface object; and

transmitting the protocol message.

83. The method as recited in claim 81, wherein managing click events of the graphical user interface object further comprises:

receiving a click event message from the client;

synchronizing logically the state of the application hierarchy to the application hierarchy of the universal client program of the client; and

triggering an un-click event for the graphical user interface object specified in the message.

84. A method for enabling operation by a client of an application program executing on a server system, comprising:

launching the application program;

managing link requests; and

managing the application program,

wherein managing the application program further comprises:

maintaining a first object-oriented hierarchy representing user interface state by the application program; and

synchronizing dynamically the first object-oriented hierarchy and a second object-oriented hierarchy representing user interface state by a universal client program by means of an event-driven synchronization protocol.

85. The method as recited in claim 84, wherein synchronizing dynamically the object-oriented hierarchies recited further comprises:

receiving user input events from the universal client program to the application program; and

communicating user interface updates from the application program to the universal client software, excluding updates known by the application program to have already been performed independently by the universal client program based on a shared knowledge of universal object logic.

86. A computer-readable medium having computer-executable instructions to cause a client computer to perform a method comprising:

launching the application program;

managing link requests; and

managing the application program,

wherein managing the application program further comprises:

maintaining a first object-oriented hierarchy representing user interface state by the application program; and

synchronizing dynamically the first object-oriented hierarchy and a second object-oriented hierarchy representing user interface state by a universal client program by means of an event-driven synchronization protocol.

87. The computer-readable medium as recited in claim 86, wherein synchronizing dynamically the object-oriented hierarchies recited further comprises:

receiving user input events from the universal client program to the application program; and

communicating user interface updates from the application program to the universal client software, excluding updates known by the application program to have already been performed independently by the universal client program based on a shared knowledge of universal object logic.

88. A computer data signal embodied in a carrier wave and representing a sequence of instructions which, when executed by a processor, cause the processor to perform the method of:

launching the application program;

managing link requests; and

managing the application program,

wherein managing the application program further comprises:

maintaining a first object-oriented hierarchy representing user interface state by the application program; and

synchronizing dynamically the first object-oriented hierarchy and a second object-oriented hierarchy representing user interface state by a universal client program by means of an event-driven synchronization protocol.

89. The computer data signal as recited in claim 88, wherein synchronizing dynamically the object-oriented hierarchies further comprises:

receiving user input events from the universal client program to the application program; and

communicating user interface updates from the application program to the universal client software, excluding updates known by the application program to have already been performed independently by the universal client program based on a shared knowledge of universal object logic.